Dynamic Deferral of Workload for Capacity Provisioning in Data Centers

Muhammad Abdullah Adnan*, Yan Ma†, Ryo Sugihara* and Rajesh Gupta*
*Department of Computer Science and Engineering
University of California, San Diego, CA, USA
Email: {madnan, ryo, rgupta}@ucsd.edu
†School of Computer Science and Technology
Shandong University, Jinan Shandong, China
Email: yam002@eng.ucsd.edu

***Abstract*—Recent increase in energy prices has led researchers to find better ways for capacity provisioning in data centers to reduce energy wastage due to the variation in workload. This paper explores the opportunity for cost saving and proposes a novel approach for capacity provisioning under bounded latency requirements for the workload. We investigate how many servers to be kept active and how much workload to be delayed for energy saving while meeting every deadline. We present an offline LP formulation for capacity provisioning by dynamic deferral and give two online algorithms to determine the capacity of the data center and the assignment of workload to servers dynamically. We prove the feasibility of the online algorithms and show that their worst case performance are bounded by a constant factor with respect to the offline formulation. We validate our algorithms on synthetic workload generated from two real HTTP traces and show that they actually perform much better in practice than the worst case, resulting in 20-40% cost-savings.**

I. Introduction

With the advent of cloud computing, data centers are emerging all over the world and their energy consumption becomes significant; as estimated 61 million MWh, ∼1.5% of US electricity consumption, costing about 4.5 billion dollars [1]. Naturally, energy efficiency in data centers has been pursued in various ways including the use of renewable energy [2], [3] and improved cooling efficiency [4], [5], [6], etc. Among them, improved scheduling algorithm is a promising approach for its broad applicability regardless of hardware configurations. While there are a number of work in this approach as well (e.g., [6], [7]), one non-conventional perspective is to optimize the schedule such that certain performance metric satisfies a predetermined requirement, which is normally defined in the form of service level agreements (SLAs). Specifically, latency is an important performance metric for any web-based services and is of great interests for service providers who run their services on data centers.

In this paper, we are interested in minimizing the energy consumption of data center under guarantees on latency/deadline. We use the deadline information to defer some tasks so that we can reduce the total cost for energy consumption for executing the workload and switching the state of the servers. We determine the portion of the released workload to be executed at the current time and the portions to be deferred to be executed at later time slots without violating deadline. Our approach is similar to 'valley filling' that is widely used in data centers to utilize server capacity during the periods of low loads [7]. But the load that is used for valley filling is mostly background/maintenance tasks (e.g. web indexing, data backup) which is different from actual workload. In fact current valley filling approaches ignore the workload characteristics for capacity provisioning. In this paper, we determine how much work to store for valley filling in order to reduce the current and future energy consumption. Later we generalize our approach for more general workload where different workload have different deadline.

The contribution of this paper is twofold. First, we present an LP formulation for capacity provisioning with dynamic deferral of workload. The formulation not only determines capacity but also determines the assignment of workload for each time slot. As a result the utilization of each server can be determined easily and resource can be allocated accordingly. Therefore this method well adapts to other scheduling policies that take into account dynamic resource allocation, priority aware scheduling, etc.

Second, we design two optimization based online algorithms depending on the nature of the deadline. For uniform deadline, our algorithm named *Valley Filling with Workload (VFW(δ))*, looks ahead δ slots to optimize the total energy consumption. The algorithm uses the valley filling approach to accumulate some workload to execute in the periods of low loads. For nonuniform deadline, we design a *Generalized Capacity Provisioning (GCP)* algorithm that reduces the switching (on/off) of servers by balancing the workloads in adjacent time slots and thus reduces energy consumption. We prove the feasibility of the solutions and show that the performance of the online algorithms are bounded by a constant factor with respect to the offline formulation in worst case. Since for the proof we do not presume anything about the workload, the performance of both the algorithms are much better in practice than the worst case, as shown by experiments. We used HTTP traces as examples for dynamic workload and found more than 40% total cost saving for GCP and around 20% total cost saving for VFW(δ) even for small deadline requirements. We compared the two online algorithms with different parameter settings and found that GCP gives more cost savings than VFW(δ) for

typical workload but for bursty workload, VFW(δ) sometimes performs better than GCP.

The rest of the paper is organized as follows. Section II presents the model that we use to formulate the optimization and gives the offline formulation. In section III, we present the VFW(δ) algorithm for determining capacity and workload assignment dynamically when the deadline is uniform. In section IV, we illustrate the GCP algorithm with nonuniform deadline. Section V shows the experimental results. In section VI, we describe the state of the art research related to capacity provisioning and section VII concludes the paper.

II. Model Formulation

In this section, we describe the model we use for capacity provisioning via dynamic deferral. The assumptions used in this model are minimal and this formulation captures many properties of current data center capacity and workload characteristics.

A. Workload Model

We consider a workload model where the total workload varies over time. The time interval we are interested in is $t \in \{0, 1, \ldots, T\}$ where T can be arbitrarily large. In practice, T can be a year and the length of a time slot τ could be as small as 2 minutes (the minimum time required to change power state of a server). In our model, the jobs have length less than τ and each job has deadline D associated with it within which it needs to be executed. If the length of a job is greater than τ then we can safely decompose it into small pieces ($\leq \tau$) each of which has deadline D. Hence we do not distinguish each job, rather deal with the total amount of workload. For now, assume that the deadline is uniform for all the workload and the non-uniform case is considered in section IV. Let L_t be the amount of workload released at time slot t. This amount of work must be executed by the end of time slot $t + D$. Since L_t varies over time, we often refer to it as a *workload curve*.

In our model, we consider a data center as a collection of homogeneous servers. The total number of servers M is fixed and given but each server can be turned on/off to execute the workload. We normalize L_t by the processing capability of each server i.e. L_t denotes the number of servers required to execute the workload at time t. We assume for all t, $L_t \leq M$. Let $x_{i,d,t}$ be the portion of the released workload L_t that is assigned to be executed at server i at time slot $t + d$ where $0 \leq d \leq D$. Let m_t be the number of active servers during time slot t. Then

$$\sum_{i=1}^{m_t} \sum_{d=0}^{D} x_{i,d,t} = L_t \text{ and } 0 \leq x_{i,d,t} \leq 1$$

Let $x_{i,t}$ be the total workload assigned at time t to server i and x_t be the total assignment at time t. Then we can think of $x_{i,t}$ as the utilization of the ith server at time t i.e. $0 \leq x_{i,t} \leq 1$. Thus

$$\sum_{d=0}^{D} x_{i,d,t-d} = x_{i,t} \text{ and } \sum_{i=1}^{m_t} x_{i,t} = x_t$$

From the data center perspective, we focus on two important decisions during each time slot t: (i) determining m_t, the number of active servers, and (ii) determining $x_{i,d,t}$, assignment of workload to the servers.

B. Cost Model

The goal of this paper is to minimize the cost (price) of energy consumption in data centers. The energy cost function consists of two parts: operating cost and switching cost. *Operating cost* is the cost for executing the workload which in our model is proportional to the assigned workload. We use the common model for the energy cost for typical servers which is an affine function:

$$C(x) = e_0 + e_1 x$$

where e_0 and e_1 are constants (e.g. see [8]) and x is the assigned workload (utilization) of a server at a time slot.

Switching cost β is the cost incurred for changing state (on/off) of a server. We consider the cost of both turning on and turning off a server. Switching cost at time t is defined as follows:

$$S_t = \beta |m_t - m_{t-1}|$$

where β is a constant (e.g. see [7], [9]).

C. Optimization Problem

Given the models above, the goal of a data center is to choose the number of active servers (capacity) m_t and the dispatching rule $x_{i,d,t}$ to minimize the total cost during $[1, T]$, which is captured by the following optimization:

$$\begin{aligned}
\min_{x_t, m_t} \quad & \sum_{t=1}^{T} \sum_{i=1}^{m_t} C(x_{i,t}) + \beta \sum_{t=1}^{T} |m_t - m_{t-1}| && (1)\\
\text{subject to} \quad & \sum_{i=1}^{m_t} \sum_{d=0}^{D} x_{i,d,t} = L_t && \forall t \\
& \sum_{i=1}^{m_t} \sum_{d=0}^{D} x_{i,d,t-d} \leq m_t && \forall t \\
& \sum_{d=0}^{D} x_{i,d,t-d} \leq 1 && \forall i, \forall t \\
& 0 \leq m_t \leq M && \forall t \\
& x_{i,d,t} \geq 0 && \forall i, \forall d, \forall t.
\end{aligned}$$

Since the servers are identical, we can simplify the problem by dropping the index i for x. More specifically, for any feasible solution $x_{i,d,t}$, we can make another solution by $x_{i,d,t} = \sum_{i=1}^{m_t} x_{i,d,t}/m_t$ (i.e., replacing every $x_{i,d,t}$ by the average of $x_{i,d,t}$ for all i) without changing the value of the objective function while satisfying all the constraints after this conversion. Then we have the following optimization equivalent to (1):

$$\min_{x_t,m_t} \quad \sum_{t=1}^{T} m_t C(x_t/m_t) + \beta \sum_{t=1}^{T} |m_t - m_{t-1}| \qquad (2)$$

$$\text{subject to} \quad \sum_{d=0}^{D} x_{d,t} = L_t \qquad \forall t$$

$$\sum_{d=0}^{D} x_{d,t-d} \le m_t \qquad \forall t$$

$$0 \le m_t \le M \qquad \forall t$$

$$x_{d,t} \ge 0 \qquad \forall d, \forall t.$$

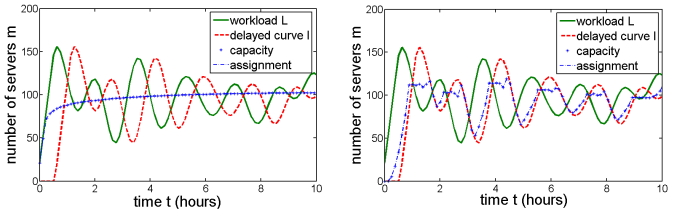
where $x_{d,t}$ represents the portion of the workload L_t to be executed at a server at time $t+d$. We further simplify the problem by showing that any optimal assignment for (2) can be converted to an equivalent assignment that uses earliest deadline first (EDF) policy. More formally, we have the following lemma:

Lemma *1:* Let $x^*_{t_r}$ and $x^*_{t_s}$ be the optimal assignments of workload obtained from the solution of optimization (2) at times t_r and t_s respectively where $t_s > t_r$ and $t_s - t_r = \theta < D$. If $\exists \delta$ with $\sum_{d=0}^{\delta-1} x^*_{d,t_r-d} \neq 0$ and $\sum_{d=\theta+\delta+1}^{D} x^*_{d,t_s-d} \neq 0$ for any $0 < \delta < D-\theta$ then we can obtain another assignments $x^e_{t_r} = x^*_{t_r}$ and $x^e_{t_s} = x^*_{t_s}$ where $\sum_{d=0}^{\delta-1} x^e_{d,t_r-d} = 0$ and $\sum_{d=\theta+\delta+1}^{D} x^e_{d,t_s-d} = 0$.

Proof: We prove it by constructing $x^e_{t_r}$ and $x^e_{t_s}$ from $x^*_{t_r}$ and $x^*_{t_s}$. We change the assignments x^*_{d,t_r}, $0 \le d \le D-\theta$ and x^*_{d,t_s}, $\theta \le d \le D$ to obtain $x^e_{t_r}$ and $x^e_{t_s}$. We now determine δ. Note that all the workloads released between (including) time slots $t_s - D$ to t_r can be executed at time t_r without violating deadline since $t_r - D < t_s - D < t_r - \delta < t_r$. Also all the workloads released between (including) time slots $t_s - D$ to t_r can be executed at time t_s without violating deadline since $t_s - D < t_r - \delta < t_r < t_s$. Hence the new assignment of workloads cannot violate any deadline. We determine δ at a point where $\sum_{d=\delta+1}^{D-\theta} x^e_{d,t_r-d} = \sum_{d=\delta+1}^{D-\theta} x^*_{d,t_r-d} + \sum_{d=\theta+\delta+1}^{D} x^*_{d,t_s-d}$ and $\sum_{d=0}^{\delta-1} x^e_{d,t_r-d} = 0$ and $x^e_{\delta,t_r-\delta} = \sum_{d=0}^{D-\theta} x^*_{d,t_r} - \sum_{d=\delta+1}^{D-\theta} x^e_{d,t_r-d}$ such that $x^e_{t_r} = x^*_{t_r}$. Similarly for $x^e_{t_s}$, we have the new assignment as: $\sum_{d=\theta}^{\theta+\delta-1} x^e_{d,t_s-d} = \sum_{d=0}^{\delta-1} x^*_{d,t_r-d} + \sum_{d=\theta}^{\theta+\delta-1} x^*_{d,t_s-d}$ and $\sum_{d=\theta+\delta+1}^{D} x^e_{d,t_s-d} = 0$ and $x^e_{\theta+\delta,t_s-\theta-\delta} = \sum_{d=\theta}^{D} x^*_{d,t_s} - \sum_{d=\theta}^{\theta+\delta-1} x^e_{d,t_s-d}$ such that $x^e_{t_s} = x^*_{t_s}$. ■

According to lemma 1, we do not need both t and d as indices of x. We can use the release time t to determine the deadline $t+D$. Thus, we drop the index d of x. At time t, unassigned workload from L_{t-D} to L_t is executed according to EDF policy while minimizing the objective function. To formulate the constraint that no assignment violates any deadline we define delayed workload l_t with maximum deadline D.

$$l_t = \begin{cases} 0 & \text{if } t \le D, \\ L_{t-D} & \text{otherwise.} \end{cases}$$



(a) Offline optimal (b) VFW(δ)

Fig. 1. Illustration of (a) offline optimal solution and (b) VFW(δ) for arbitrary workload generated randomly; time slot length = 2 min, $D = 15$, $\delta = 10$.

We call the delayed curve l_t for the workload as *deadline curve*. Thus we have two fundamental constraints on the assignment of workload for all t:

(C1) Deadline Constraint: $\sum_{j=1}^{t} l_j \le \sum_{j=1}^{t} x_j$

(C2) Release Constraint: $\sum_{j=1}^{t} x_j \le \sum_{j=1}^{t} L_j$

Condition (C1) says that all the workloads assigned up to time t cannot violate deadline and Condition (C2) says that the assigned workload up to time t cannot be greater than the total released workload up to time t. Using these constraints we reformulate the optimization (2) as follows:

$$\min_{x_t,m_t} \quad \sum_{t=1}^{T} m_t C(x_t/m_t) + \beta \sum_{t=1}^{T} |m_t - m_{t-1}| \qquad (3)$$

$$\text{subject to} \quad \sum_{j=1}^{t} l_j \le \sum_{j=1}^{t} x_j \le \sum_{j=1}^{t} L_j \qquad \forall t$$

$$\sum_{j=1}^{T} x_j = \sum_{j=1}^{T} L_j$$

$$0 \le x_t \le m_t \qquad \forall t$$

$$0 \le m_t \le M \qquad \forall t.$$

Since the operating cost function $C(.)$ is an affine function, the objective function is linear as well as the constraints. Hence it is clear that the optimization (3) is a linear program. Note that capacity m_t in this formulation is not constrained to be an integer. This is acceptable because data centers consists of thousands of active servers and we can round the resulting solution with minimal increase in cost. Figure 1(a) illustrates the offline optimal solutions for x_t and m_t for a dynamic workload generated randomly. The performance of the optimal offline algorithm on two realistic workload are provided in Section V.

III. Valley Filling with Workload

In this section we consider the online case, where at any time t, we do not have information about the future workload $L_{t'}$ for $t' > t$. At each time t, we determine the x_t and m_t by applying optimization over the already released unassigned workload which has deadline in future D slots. Note that the workload released at or before t, can not be delayed to be assigned after time slot $t+D$. Hence we do not optimize over more than $D+1$ slots. We simplify the online optimization

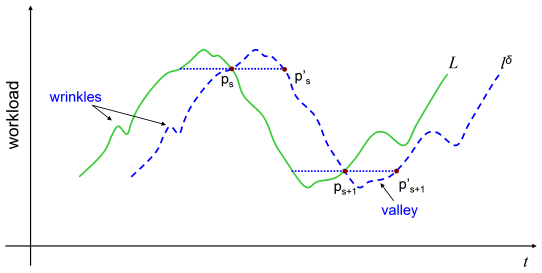


Fig. 2. The curves L_t and l_t^δ and their intersection points.

by solving only for m_t and determine x_t by making $x_t = m_t$ at time t. This makes the online algorithm not to waste any execution capacity that cannot be used later for executing workload. But the cost due to switching in the online algorithm may be higher than the offline algorithm. Thus our goal is to design strategies to reduce the switching cost. In the online algorithm, we reduce the switching cost by optimizing the total cost for the interval $[t, t + D]$.

When the deadline is uniform, we can reduce the switching cost even more by looking beyond D slots. We do that by accumulating some workload from periods of high loads and execute that workload later in valleys without violating constraints (C1) and (C2). To determine the amount of accumulation and execution we use 'δ-delayed workload'. Thus the online algorithm namely Valley Filling with Workload (VFW(δ)) looks ahead δ slots to determine the amount of execution. Let l_t^δ be the *δ-delayed curve* with delay of δ slots for $0 < \delta < D$.

$$l_t^\delta = \begin{cases} 0 & \text{if } t \le \delta, \\ L_{t-\delta} & \text{otherwise.} \end{cases}$$

Then we can call the deadline curve as *D-delayed curve* and represent it by l_t^D. We determine the amount of accumulation and execution by controlling the set of feasible choices for m_t in the optimization. For this we use the δ-delayed curve to restrict the amount of accumulation. By lower bounding m_t for the valley (low workload) and upper bounding it for the high workload, we control the execution in the valley and accumulation in the other parts of the curve. In the online algorithm we have two types of optimizations: Local Optimization and Valley Optimization. Local Optimization is used to smooth the 'wrinkles' (small variation in the workload in adjacent slots e.g. see Figure 2) within D consecutive slots and accumulate some workload and Valley Optimization fills the valleys with the accumulated workload.

A. Local Optimization

The *local optimization* applies optimization over future D slots and finds the optimum capacity for current slot by executing not more than δ-delayed workload. Let t be the current time slot. At this slot we apply a slightly modified version of offline optimization (3) in the interval $[t, t + D]$. Then we apply the following optimization LOPT(l_t, l_t^δ, m_{t-1}, M) to determine m_t in order to smooth the wrinkles by optimizing over D consecutive slots. We restrict the amount of execution to be no more than the δ-delayed workload while satisfying the deadline constraint (C1).

$$\begin{aligned} \min_{m_t} \quad & (e_0 + e_1) \sum_{j=t}^{t+D} m_j + \beta \sum_{j=t}^{t+D} |m_j - m_{j-1}| \qquad (4) \\ \text{subject to} \quad & \sum_{j=1}^{t} l_j^D \le \sum_{j=1}^{t} m_j \\ & \sum_{j=1}^{t+D} m_j = \sum_{j=1}^{t+\delta} l_j^\delta \\ & 0 \le m_k \le M \qquad\qquad t \le k \le t + D \end{aligned}$$

After solving the local optimization, we get the value of m_t for the current time slot and assign $x_t = m_t$. For the next time slot $t + 1$ we solve the local optimization again to find the values for x_{t+1} and m_{t+1}. Note that the deadline constraint (C1) and the release constraint (C2) are satisfied at time t, since from the formulation $\sum_{j=1}^{t} l_j^D \le \sum_{j=1}^{t} m_j \le \sum_{j=1}^{t} l_j^\delta \le \sum_{j=1}^{t} L_j$.

B. Valley Optimization

In *valley optimization*, the accumulated workload from the local optimization is executed in 'global valleys'. Before giving the formulation for the valley optimization we need to detect a valley.

Let $p_1, p_2, \ldots, p_n$ be the sequence of intersection points of L_t and l_t^δ curves (see Figure 2) in nondecreasing order of their x-coordinates (t values). Let $p'_1, p'_2, \ldots, p'_n$ be the sequence of points on l_t^δ with delay δ added with each intersection point $p_1, p_2, \ldots, p_n$ on l_t^δ such that $t'_s = t_s + \delta$ for all $1 \le s \le n$. We discard all the intersection points (if any) between p_s and p'_s from the sequence such that $t_{s+1} \ge t'_s$. Note that at each intersection point p_s, the curve from p_s to p'_s is known. To determine whether the curve l_t^δ between p_s and p'_s is a valley, we calculate the area

$$A = \sum_{t=t_s}^{t'_s} (l_t^\delta - l_{t_s}^\delta)$$

If A is negative, then we regard the curve between p_s and p'_s as a *global valley* though it may contain several peaks and valleys. If the curve between p_s and p'_s is a global valley, we fill the valley with some (possibly all) of the accumulated workload by executing more than the δ-delayed workload while satisfying the release constraint (C2). For each t, we apply the following optimization VOPT(l_t, L_t, m_{t-1}, M) in the interval $[t, t+D]$ to find the value of m_t where $t_s \le t \le t'_s$.

$$\min_{m_t} \quad (e_0+e_1)\sum_{j=t}^{t+D} m_j + \beta \sum_{j=t}^{t+D} |m_j - m_{j-1}| \qquad (5)$$

$$\text{subject to} \quad \sum_{j=1}^{t} l_j^D \le \sum_{j=1}^{t} m_j$$

$$\sum_{j=1}^{t+D} m_j = \sum_{j=1}^{t} L_j$$

$$0 \le m_k \le M \qquad t \le k \le t+D$$

Note that the deadline constraint (C1) and the release constraint (C2) are satisfied at time t, since $\sum_{j=1}^{t} l_j^D \le \sum_{j=1}^{t} m_j \le \sum_{j=1}^{t} L_j$. We apply the valley optimization (5) for each $t_s \le t \le t'_s$ and local optimization (4) for each time slot t where $t \in \{[1, T-D-1] - [t_s, t'_s]\}$ for all t_s. For each $t \in [T-D, T]$ we apply the valley optimization (5) for global valley in the interval $[t, T]$ in order to execute all the accumulated workload. Algorithm 1 summarizes the procedures for VFW(δ). For each new time slot t, Algorithm 1 detects a valley by checking whether the curves l_t^δ and L_t intersects. If t is inside a valley, Algorithm 1 applies valley optimization (VOPT); local optimization (LOPT), otherwise. Figure 1(b) illustrates the nature of solutions from VFW(δ) for x_t and m_t. Note that δ is a parameter for the online algorithm VFW(δ).

Algorithm 1 VFW(δ)

```
valley ← 0; m_0 ← 0
l^D[1 : D] ← 0; l^δ[1 : δ] ← 0
for each new time slot t do
    l^D[t + D] ← L[t]
    l^δ[t + δ] ← L[t]
    if valley = 0 and l^δ intersects L then
        Calculate Area A
        if A < 0 then
            valley ← 1
        end if
    else if valley > 0 and valley ≤ δ then
        valley ← valley + 1
    else
        valley ← 0
    end if
    if valley = 0 then
        m[t : t+D] ← LOPT(l[1 : t], l^δ[1 : t+δ], m_{t−1},M)
    else
        m[t : t + D] ← VOPT(l[1 : t], L[1 : t], m_{t−1}, M)
    end if
    x_t ← m_t
end for
```

C. Analysis of the Algorithm

We first prove the feasibility of the solutions from the VFW(δ) algorithm and then analyze the competitive ratio of this algorithm with respect to the offline formulation (3). First, we have the following theorem about the feasibility.

Theorem *1:* The VFW(δ) algorithm gives feasible solution for any $0 < \delta < D$.

Proof: We prove this theorem inductively by showing that the choice of any feasible m_t from an optimization applied in the interval $[t, t+D]$ do not result in infeasibility in the optimization applied in $[t+1, t+D+1]$. Initially, the optimization in VFW(δ) is applied for the interval $[1, D+1]$ with $\sum_{j=1}^{k} l_j^D = 0$ for $1 \le k \le D$. Hence the optimization applied in the intervals $[1, D+1]$ gives feasible m_1 because $\sum_{j=1}^{k} l_j^D \le \sum_{j=1}^{k} l_j^\delta \le \sum_{j=1}^{k} L_j$ for $1 \le k \le D$.

Now suppose the VFW(δ) gives feasible m_t in an interval $[t, t+D]$. We have to prove that there exists feasible choice for m_t for the optimization applied at $[t+1, t+D+1]$. The deadline constraint (C1) and the release constraint (C2) are satisfied for m_t. Hence, $\sum_{j=1}^{t} l_j^D \le \sum_{j=1}^{t} l_j^\delta \le \sum_{j=1}^{t} L_j$. Since $0 < \delta < D$, $\sum_{j=1}^{t} l_j^D \le \sum_{j=1}^{t+1} l_j^D \le \sum_{j=1}^{t} l_j^\delta \le \sum_{j=1}^{t+1} l_j^\delta \le \sum_{j=1}^{t} L_j \le \sum_{j=1}^{t+1} L_j$. Thus for any feasible choice of m_t, we can always obtain feasible solution for m_{t+1} such that the above inequality holds. ■

We now analyze the competitive ratio of the online algorithm with respect to the offline formulation (3). We denote the operating cost of the solution vectors $X = (x_1, x_2, \ldots, x_T)$ and $M = (m_1, m_2, \ldots, m_T)$ by $cost_o(X, M) = \sum_{t=1}^{T} m_t C(x_t/m_t)$, switching cost by $cost_s(X, M) = \beta \sum_{t=1}^{T} |m_t - m_{t-1}|$ and total cost by $cost(X, M) = cost_o(X, M) + cost_s(X, M)$. We have the following lemma.

Lemma *2:* $cost_s(X, M) \le 2\beta \sum_{t=1}^{T} m_t$

Proof: Switching cost at time t is $S_t = \beta|m_t - m_{t-1}| \le \beta(m_t + m_{t-1})$, since $m_t \ge 0$. Then $cost_s(X, M) \le \beta \sum_{t=1}^{T}(m_t + m_{t-1}) \le 2\beta \sum_{t=1}^{T} m_t$ where $m_0 = 0$. ■

Let X^* and M^* be the offline solution vectors from optimization (3). We have the following theorem about the competitive ratio.

Theorem *2:* $cost(X, M) \le \frac{e_0+e_1+2\beta}{e_0+e_1} cost(X^*, M^*)$.

Proof: Since the offline optimization assigns all the workload in the $[1, T]$ interval, $\sum_{t=1}^{T} x_t^* = \sum_{t=1}^{T} L_t \le \sum_{t=1}^{T} m_t^*$, where we used $x_t^* \le m_t^*$ for all t. Hence $cost(X^*, M^*) \ge cost_o(X^*, M^*) = \sum_{t=1}^{T} m_t^* C(x_t^*/m_t^*) = \sum_{t=1}^{T}(e_0 m_t^* + e_1 x_t^*) \ge \sum_{t=1}^{T}(e_0 + e_1) L_t$.

In the online algorithm we set $x_t = m_t$ and $\sum_{j=1}^{t} m_j \le \sum_{j=1}^{t} L_j$ for all $t \in [1, T]$. Hence by lemma 2, we have $cost(X, M) = cost_o(X, M) + cost_s(X, M) \le \sum_{t=1}^{T}(e_0 + e_1) m_t + 2\beta \sum_{t=1}^{T} m_t \le (e_0 + e_1) \sum_{t=1}^{T} L_t + 2\beta \sum_{t=1}^{T} L_t = (e_0 + e_1 + 2\beta) \sum_{t=1}^{T} L_t$. ■

Note that the competitive ratio does not depend on δ or D. Hence the performance of the VFW(δ) is within a constant factor of the offline algorithm. Although the ratio seems to be large, the performance of VFW(δ) algorithm is close to the offline optimal algorithm as evaluated in section V.

IV. Generalized Capacity Provisioning

We now consider the general case where the deadline requirement is not same for all the workload. Let ν be the maximum possible deadline. We decompose the workload according to their associated deadline. Suppose $L_{d,t} \geq 0$ be the portion of the workload released at time t and has deadline d for $0 \leq d \leq \nu$. We have

$$\sum_{d=0}^{\nu} L_{d,t} = L_t$$

The workload to be executed at any time slot t can come from different previous slots $t-d$ where $0 \leq d \leq \nu$ as illustrated in Figure 3(a). Hence we redefine the deadline curve l_t and represent it by l'_t. Assuming $L_{d,t} = 0$ if $t \leq 0$, we define

$$l'_t = \sum_{d=0}^{\nu} L_{d,(t-d)}$$

Then the offline formulation remains the same as formulation (3) with the deadline curve l_t replaced by l'_t.

$$\begin{aligned}
\min_{x_t, m_t} \quad & \sum_{t=1}^{T} m_t C(x_t/m_t) + \beta \sum_{t=1}^{T} |m_t - m_{t-1}| && (6)\\
\text{subject to} \quad & \sum_{j=1}^{t} l'_j \leq \sum_{j=1}^{t} x_j \leq \sum_{j=1}^{t} L_j && \forall t\\
& \sum_{j=1}^{T} x_j = \sum_{j=1}^{T} L_j \\
& 0 \leq x_t \leq m_t && \forall t\\
& 0 \leq m_t \leq M && \forall t.
\end{aligned}$$

We now consider the online case. Delaying the workload up to their maximum deadline may increase the switching cost since it may increase the variation in the workload compared to the original workload (see Figure 3(b)). Hence at each time we need to determine the optimum assignment and capacity that reduces the switching cost from the original workload while satisfying each individual deadline. We can apply the VFW(δ) algorithm from the previous section with $D = D_{min}$ where D_{min} is the minimum deadline for the workload. If D_{min} is small, VFW(δ) does not work well because $\delta < D_{min}$ becomes too small to detect a valley. Hence we use a novel approach for distributing the workload L_t over the D_t slots such that the change in the capacity between adjacent time slots is minimal (see Figure 3(c)). We call this algorithm Generalized Capacity Provisioning (GCP) algorithm.

In the GCP algorithm, we apply optimization to determine m_t at each time slot t and make $x_t = m_t$. The optimization is applied over the interval $[t, t+\nu]$ since at time slot t we can have workload that has deadline up to $t+\nu$ slots. Hence at each time t, the released workload is a vector of $\nu+1$ dimension. Let, $\mathbf{L_t} = (L_{0,t}, L_{1,t}, \ldots, L_{\nu,t})$ where $L_{d,t} = 0$ if there is no workload with deadline d at time t. Let $\mathbf{y_t}$ be the vector of unassigned workload released up to time t. The vector $\mathbf{y_t}$ is updated from $\mathbf{y_{t-1}}$ at each time slot by subtracting the capacity m_{t-1} and then adding $\mathbf{L_t}$. Note that m_{t-1} is subtracted from the vector $\mathbf{y_{t-1}}$ in order to use unused capacity to execute already released workload at time $t-1$ by following EDF policy (see lines 4-17 in Algorithm 2). Let $\mathbf{y'_{t-1}} = (y'_{0,t-1}, y'_{1,t-1}, y'_{2,t-1}, \ldots, y'_{\nu,t-1})$ be the vector after subtracting m_{t-1} with $y'_{0,t-1} = 0$ and $y'_{j,t-1} \geq 0$ for $1 \leq j \leq \nu$. Then $\mathbf{y_t} = (y'_{1,t-1}, y'_{2,t-1}, \ldots, y'_{\nu,t-1}, 0) + \mathbf{L_t}$ where $\mathbf{y_t} = (0, 0, \ldots, 0)$ if $t <= 0$. Then the optimization GCP-OPT($\mathbf{y_t}$, m_{t-1}, M) applied at each t over the interval $[t, t+\nu]$ is as follows:

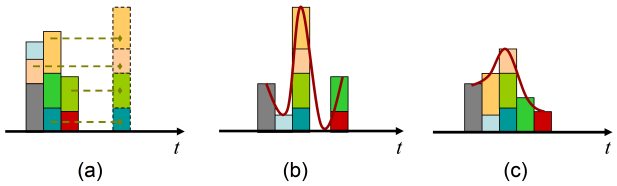


Fig. 3. Illustration of workload with different deadline requirements. (a) workload released at different times have different deadlines, (b) the delayed workload l'_t, may increase the switching cost due to large variation, (c) distribution of workload in adjacent slots by GCP to reduce the variation in workload.

$$\begin{aligned}
\min_{m_t} \quad & (e_0 + e_1) \sum_{j=t}^{t+\nu} m_j + \beta \sum_{j=t}^{t+\nu} |m_j - m_{j-1}| && (7a)\\
\text{subject to} \quad & \sum_{j=0}^{\nu} m_{t+j} = \sum_{j=0}^{\nu} y_{j,t} && (7b)\\
& \sum_{k=0}^{j} m_{t+k} \geq \sum_{k=0}^{j} y_{k,t} && 0 \leq j \leq \nu - 1 \quad (7c)\\
& 0 \leq m_{t+j} \leq M && 0 \leq j \leq \nu \quad (7d)
\end{aligned}$$

Note that the optimization (7) solves for $\nu+1$ values. We only use m_t as the capacity and assignment of workload at time t. Algorithm 2 summarizes the procedures for GCP. The GCP algorithm gives feasible solutions because it works with the unassigned workload and constraint (7c) ensures deadline constraint (C1) and constraint (7b) ensures the release constraint (C2). The competitive ratio for the GCP algorithm is same as the competitive ratio for VFW(δ) because in GCP, $m_t = x_t$ and release constraint (C2) holds at every t making $\sum_{t=1}^{T} m_t = \sum_{t=1}^{T} x_t \leq \sum_{t=1}^{T} L_t$.

V. Experimental Results

In this section, we seek to evaluate the cost incurred by the VFW(δ) and GCP algorithm relative to optimal solution in the context of workload generated from realistic data.

A. Experimental Setup

We aim to use realistic parameters in the experimental setup and provide conservative estimates of cost savings resulting from our proposed VFW(δ) and GCP algorithms.

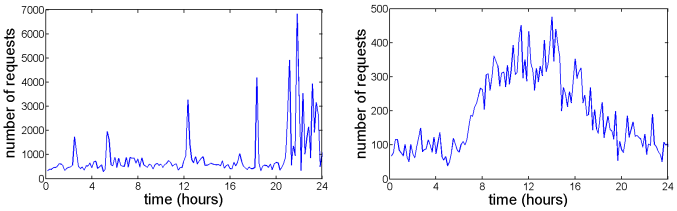
Algorithm 2 GCP

$y[0:\nu] \leftarrow 0$
$m_0 \leftarrow 0$
for each new time slot t **do**
  $uc \leftarrow m_{t-1}$ {uc represents the unused capacity}
  for $i = 0$ to ν **do**
    if $uc \leq 0$ **then**
      $y'[i] \leftarrow y[i]$
    else
      $uc \leftarrow uc - y[i]$
      if $uc \leq 0$ **then**
        $y'[i] \leftarrow -uc$
      else
        $y'[i] \leftarrow 0$
      end if
    end if
  end for
  $y[0:\nu] = \{y'[1:\nu], 0\} + L_t[0:\nu]$
  $m[t:t+D] \leftarrow$ GCP-OPT($y[0:\nu]$, m_{t-1}, M)
  $x_t \leftarrow m_t$
end for

Cost benchmark: Currently data centers typically do not use dynamic capacity provisioning based on the variation of the workload [7]. A naive approach for capacity provisioning is to follow the workload curve and determine the capacity and assignment of workload accordingly. Clearly it is not a good approach because for capacity provisioning it does not take into account the cost incurred due to switching. Yet this is a very conservative estimate as it does not waste any execution capacity and meets all the deadline. We compare the total cost from the VFW(δ) and GCP algorithm with the 'follow the workload' ($x = m = L$) strategy and evaluate the cost reduction.

Cost function parameters: The total cost is characterized by e_0 and e_1 for the operating cost and β for the switching cost. In the operating cost, e_0 represents the proportion of the fixed cost and e_1 represents the load dependent energy consumption. The energy consumption of the current servers is dominated by the fixed cost [10]. Therefore we choose $e_0 = 1$ and $e_1 = 0$. The switching cost parameter β represents the wear-and-tear due to changing power states in the servers. We choose $\beta = 6$ for slot length of 10 minutes such that it works as an estimate of the time a server should be powered down (typically one hour [7], [9]) to outweigh the switching cost with respect to the operating cost.

Workload description: We use two HTTP traces from real world as examples of dynamic workload. The HTTP traces are taken from the HTTP request logs for one day (24 hours) from a server at University of California San Diego (UCSD) and San Diego Super Computing Center (SDSC). We counted the number of different types of requests over a time slot length of 10 minutes and use that as a dynamic workload (Figure 4). The two examples we use represent strong diurnal properties and have variation from bursty workload (UCSD) to typical workload (SDSC). We then assign deadline for each workload. For VFW(δ), the deadline D is uniform and is assigned in terms of number of slots the workload can be delayed. For our experiments, We vary D from $1-15$ slots which gives latency from 10 minutes upto 2 hour 30 minutes. Note that the choice for the values of D are intended for the synthetic dynamic workload and not for real http requests. For GCP, we use the request types to generate workload with different deadline requirements. For same type of requests we chose the same value for d but assigned different value for different types e.g. image files, video files, text files etc. We picked ten different file types from the requests and used their relative frequency to assign deadline varying from $1-10$ slots.



(a) UCSD (b) SDSC

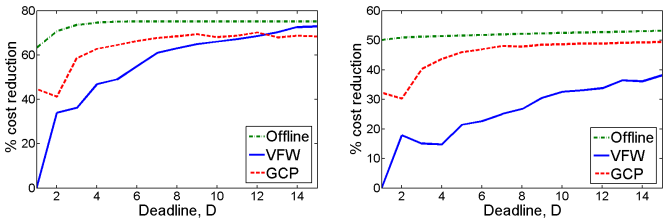
Fig. 4. Illustration of the traces for dynamic workload used in the experiments.

B. Experimental Analysis

In this section we analyze the impact of wide variety of parameters on cost savings provided by VFW(δ) and GCP. We then compare VFW(δ) and GCP for uniform deadline (GCP-U) and justify practical significance of our work.

Impact of deadline: The first parameter we study is the impact of different deadline requirements of the workload on the cost savings. Figure 5 shows that even for deadline D as small as 2 slots, the cost is reduced by $\sim$40% for GCP-U, $\sim$30% for VFW(δ) while the offline algorithm gives a cost saving of $\sim$60% compared to the naive algorithm. It also shows that for all the algorithms, large D gives more cost savings as more workload can be delayed to reduce the variation in the workload. As D grows larger the cost reduction from GCP-U and VFW(δ) approaches offline cost saving which is as much as 70%. For VFW(δ), the cost saving is always less than GCP-U for typical workload (SDSC), but for bursty workload (UCSD) VFW(δ) performs better than GCP-U as filling valleys with the workload becomes more beneficial when D becomes large.

Impact of δ for VFW(δ): The parameter δ is used as a lookahead to detect a valley in the VFW(δ) algorithm. If δ is large, valley detection performs well but it may be too late to fill the valley due to the deadlines. On the other hand if δ is small, valley detection does not work well because the capacity has already gone down to the lowest value. Figure 6 illustrates the valley detection for small δ and large δ. Although the cost savings from VFW(δ) largely depends the nature of the workload curve, Figure 7 shows that $\delta \sim D/2$ is a conservative estimate for better cost savings.



(a) UCSD (b) SDSC

Fig. 5. Impact of deadline on cost incurred by GCP-U, Offline and VFW(δ) with $\delta = D/2$.

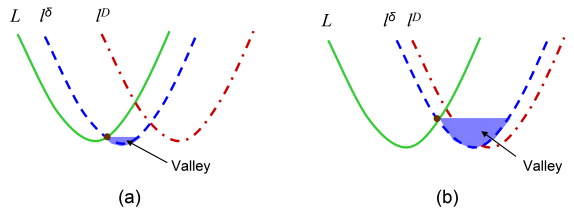
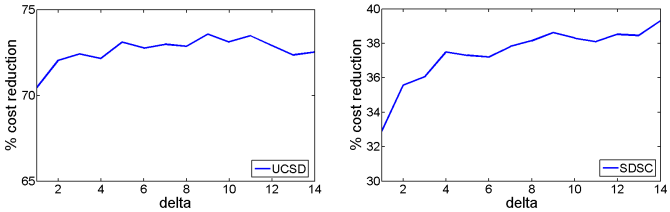


Fig. 6. Valley detection for (a) small δ and (b) large δ for VFW(δ).



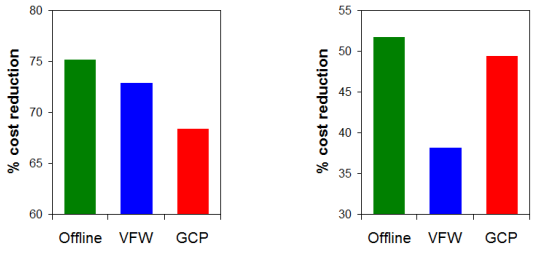
(a) UCSD (b) SDSC

Fig. 7. Impact of δ for VFW(δ) with deadline $D = 15$.

Performance of GCP: We evaluated the cost savings from GCP by assigning different deadline for different types of workload. For conservative estimates of deadline requirements, we found ∼60% cost reduction for UCSD workload and ∼40% cost reduction for SDSC workload each of which remains close to the offline optimal solutions.

Comparision of VFW(δ) and GCP: We compare GCP for uniform deadline (GCP-U) with VFW(δ) for $\delta = D/2$. Figure 8 illustrates the cost reduction for VFW(δ) and GCP-U with uniform deadline $D = 15$ and $\delta = 8$. Surprisingly for bursty workload (UCSD), VFW(δ) works much better than GCP-U for any value of δ. But for typical workload GCP-U works better. Hence the performance of both the online algorithms depends largely on the nature of the workload.

Practical significance: We now justify different practical aspects of capacity provisioning via dynamic deferral. Although we have used synthesized workload for our experiments, there are many real world examples in High Performance Computing (HPC) which has real deadline requirements e.g. see [11], [12]. Lee et al. [11] conducted a survey to measure the impact of delay on user satisfaction and represented delay as utility functions which are often flat indicating fixed deadline requirements. The deadline requirements in HPC are usually



(a) UCSD (b) SDSC

Fig. 8. Comparision of VFW(δ), GCP-U and Offline algorithms with uniform deadline $D = 15$ and $\delta = 8$.

large in orders of hours but in other applications it might be small in order of minutes. Hence we studied the impact of different deadline requirements varying from 10 minutes to 2 hours and 30 minutes (see Figure 5). Our results highlight that even if the deadline is as small as one slot (10 minutes), we save around 40% energy consumption compared to without using dynamic deferral. Note that we do not use any prediction window [7], [13] for the algorithms. Using prediction window, our algorithms can be modified to optimize over more future slots which eventually results in more energy savings.

VI. Related Work

With the importance of energy management in data centers, many scholars have applied energy-aware scheduling because of its low cost and practical applicability. In energy-aware scheduling, most work tries to find a balance between energy cost and performance loss through DVFS (Dynamic Voltage and Frequency Scaling) and DPM (Dynamic Power Management), which are the most common system-level power saving methods. Beloglazov et al. [14] give the taxonomy and survey on energy management in data centers. Dynamic capacity provisioning is part of DPM technique. Chase et al. [15] introduce the executable utility functions to quantify the value of performance and use economic approach to achieve resource provisioning. Pinheiro et al. [16] consider resource provisioning in both application and operating system level. They dynamically turn on or turn off nodes to adapt to the changing load, but do not consider the switching cost.

Most work on dynamic capacity provisioning for independent workload uses models based on queueing theory [17], [18], [19], or control theory [20], [21], [22]. Recently Lin et al. [7] used more general and common energy model and delay model which are not confined to queueing or control theoretic methods. In our paper we also use general energy model to extend its usage for latency requirements. However, our work is different from Lin et al. [7] in several ways. First, the performance of their LCP algorithm depends on the peak-to-mean ratio (PMR) of the workload, while our algorithms perform better for workloads with high PMR. Second, in LCP algorithm, smaller time slot is desirable because switch cost depends on the length of time slot, and thus the difference between upper and lower limit for the capacity increases for

LCP algorithm and its capacity curve comes closer to the workload curve resulting in higher switch cost than ours. At last, in concerning delay, LCP considers delay as the objective and aim to minimize the average delay while we regard it as the deadline constraint. Instead of penalizing the delay, we provide guarantee on maximum delay and utilize delay to reduce the switching cost of the servers.

Many applications in real world require delay bound or deadline constraint e.g. see Lee et al. [11]. When combining with energy conservation, deadline is usually a critical adjusting tool between performance loss and energy consumption. Energy efficient deadline scheduling was first studied by Yao et al. [23]. They proposed one offline algorithm and two online algorithms, which aim to minimize energy consumption for independent jobs with deadline constraints on a single variable-speed processor. After that, a series of work was done to consider online deadline scheduling in different scenarios, such as discrete-voltage processor, tree-structured tasks, processor with sleep state and overloaded system [24], [25]. There are also some work on energy-aware scheduling in multiprocessor systems. Most of them focus on real-time tasks [26], [27], [28]. In the context of data center, most work on energy management merely talk about minimizing the average delay but not give any bound on delay except Mukherjee et al. [6]. They propose an offline algorithm-SCINT and online algorithm-EDF-LRH considering deadline constraints to minimize the computation, cooling and migration energy. In the online algorithm, they simply use EDF algorithm to satisfy the deadline, while in the offline algorithm they use genetic algorithm. However, this work is a job assignment problem not a dynamic resource provisioning problem, where the number of needed servers is given in advance.

VII. Conclusion

In this paper we have proposed two new algorithms VFW(δ) and GCP for capacity provisioning in data centers while guaranteeing the deadlines. The algorithms utilize the latency requirements of workloads for cost savings and guarantees bounded cost and bounded latency under very general settings - arbitrary workload, general deadline and general energy cost models. Further both the VFW(δ) and GCP algorithms are simple to implement and do not require significant computational overhead.

Our experiments highlight that significant cost and energy savings can be achieved via dynamic deferral of workload. In this paper, we tried to limit our motivation towards data centers. But in the future 'cloudy world' where almost all the computation will be outsourced, deadline/latency requirements would catch more attention. Therefore it would be worth and interesting to apply the concept of dynamic defferal to other load balancing or scheduling problems. We keep that as our future work.

References

[1] N. Anderson, *Epa: Power usage in data centers could double by 2011*, http://arstechnica.com/old/content/2007/08/ epa-power-usage-in-data-centers-could-double-by-2011.ars, August 2007.

[2] Z. Liu, M. Lin, A. Wierman, S. Low, and L. Andrew, *G*reening Geographical Load Balancing, in Proc. SIGMETRICS, June 2011.

[3] C. Stewart and K. Shen, *Some Joules Are More Precious Than Others: Managing Renewable Energy in the Datacenter,* in Proc. Power Aware Comput. and Sys., October 2009.

[4] E. Pakbaznia and M. Pedram, *Minimizing data center cooling and server power costs*, in Proc. ISLPED, 2009.

[5] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, J. S. Chase, *Balance of Power: Dynamic Thermal Management for Internet Data Centers,* IEEE Internet Computing, vol. 9, no. 1, pp. 42-49, 2005.

[6] T. Mukherjee, A. Banerjee, G. Varsamopoulos, and S. K. S. Gupta, *Spatio-Temporal Thermal-Aware Job Scheduling to Minimize Energy Consumption in Virtualized Heterogeneous Data Centers*, Computer Networks, 2009.

[7] M. Lin, A. Wierman, L. H. Andrew, and E. Thereska, *Dynamic right-sizing for power-proportional data centers*, in Proc. IEEE INFOCOM, 2011.

[8] *SPEC power data on SPEC website at http://www.spec.org.*

[9] P. Bodik, M. P. Armbrust, K. Canini, A. Fox, M. Jordan, and D. A. Patterson, *A case for adaptive datacenters to conserve energy and improve reliability*, University of California at Berkeley, Tech. Rep. UCB/EECS-2008-127, 2008.

[10] L. A. Barroso, and U. Holzle, *The case for energy-proportional computing.* Computer, vol. 40, no. 12, pp. 3337, 2007.

[11] C. B. Lee, and A. Snavely, *Precise and realistic utility functions for user-centric performance analysis of schedulers*, In Proc. IEEE Symp. on High-Performance Distributed Computing (HPDC), June 2007.

[12] Atsuko Takefusa, Satoshi Matsuoka, Henri Casanova, Francine Berman, *A Study of Deadline Scheduling for Client-Server Systems on the Computational Grid,* in Proc. 10th IEEE Int. Sym. on HPDC, pp. 04-06, 2001.

[13] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, *Workload analysis and demand prediction of enterprise data center applications*, In Proc. IEEE Symp. Workload Characterization, Sep. 2007.

[14] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, *A taxonomy and survey of energy-efficient data centers and cloud computing systems*, Univ. of Melbourne, Tech. Rep. CLOUDS-TR-2010-3, 2010.

[15] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, *Managing energy and server resources in hosting centers*, in Proc. ACM SOSP, pp. 103-116, 2001.

[16] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath, *Load balancing and unbalancing for power and performacne in cluster-based systems*, in Proc. Compilers and Operating Systems for Low Power, 2001.

[17] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, *Optimal power allocation in server farms*, in Proc. ACM Sigmetrics, 2009.

[18] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch, *Optimality analysis of energy-performance trade-off for server farm management*, Performance Evaluation, vol. 67, no. 11, pp. 1155 - 1171, 2010.

[19] D. Meisner, B. T. Gold, and T. F. Wenisch, *The PowerNap Server Architecture*, ACM trans. Computer systems (TOCS), 29(1), 2011.

[20] T. Horvath and K. Skadron, *Multi-mode energy management for multi-tier server clusters*, in Proc. PACT, 2008.

[21] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, *Managing server energy and operational costs in hosting centers*, in Proc. ACM Sigmetrics, 2005.

[22] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely, *Dynamic resource allocation and power management in virtualized data centers,* in Proc. IEEE/IFIP NOMS, 2010.

[23] Frances Yao, Alan Demers, and Scott Shenker, *A scheduling model for reduced CPU energy*, In Proc 36th IEEE symp on foundations of computer science (FOCS), pp. 374-382, 1995.

[24] H. L. Chan, J. W. T. Chan, T. W. Lam, L. K. Lee, K. S. Mak and P. W. Wong, *Optimizing throughput and energy in online deadline scheduling*, ACM Trans. Algorithms 6(1), 1-10, 2009.

[25] X. Han, T. W. Lam, L. K. Lee, I. K. To and P. W. Wong, *Deadline scheduling and power management for speed bounded processors*, Theor. Comput. Sci. 411(40-42), 3587-3600, 2010.

[26] N. K. Jha, *Low power system scheduling, synthesis and displays*, IEE Proc. on Comput. and Digital Tech., 152(3), 34-35, 2005.

[27] K. Q. Li, *Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed*, IEEE Trans. Parall. Distr., 19(11), 1484-1497, 2008.

[28] H. Chen, A. M. K. Cheng and Y. W. Kuo, *Assigning real-time tasks to heterogeneous processors by applying ant colony optimization*, J. Parallel Distrib. Comput., 71(1), 132-142, 2011.